Towards a Neural Hierarchy of Time Scales for Motor Control

Tim Waegeman, Francis Wyffels, Benjamin Schrauwen

Department of Electronics and Information Systems
Ghent University, Ghent Belgium
http://reslab.elis.ugent.be

Abstract. Animals show remarkable rich motion skills which are still far from realizable with robots. Inspired by the neural circuits which generate rhythmic motion patterns in the spinal cord of all vertebrates, one main research direction points towards the use of central pattern generators in robots. On of the key advantages of this, is that the dimensionality of the control problem is reduced. In this work we investigate this further by introducing a multi-timescale control hierarchy with at its core a hierarchy of recurrent neural networks. By means of some robot experiments, we demonstrate that this hierarchy can embed any rhythmic motor signal by imitation learning. Furthermore, the proposed hierarchy allows the tracking of several high level motion properties (e.g.: amplitude and offset), which are usually observed at a slower rate than the generated motion. Although these experiments are preliminary, the results are promising and have the potential to open the door for rich motor skills and advanced control.

Keywords: Locomotion Control Hierarchy, Adaptive control, Feedback control, Central Pattern Generator, Reservoir computing (RC)

1 Introduction

Animals show remarkable rich motion skills, they are able to run and walk over uneven and difficult terrain without the need to think about breathing, muscle control or low level sensory feedback processing. Instead, they think on a more high level such as which obstacles are approaching and how they can avoid them.

According to [1], many aspects of brain functions can be explained by a hierarchy of temporal scales at which representations of the environment evolve. The higher level encodes slower contextual changes in the environment or body while at the lower level faster variations due to sensory processing are encoded.

Other biological research suggests that specialized neural circuits, so called central pattern generators (CPGs), located in the spinal cord are responsible for generating rhythmic activations needed for body function including the contractions of a heart or lungs and control of muscles for walking [2]. Implying that some aspects of brain functions are offloaded to regions outside of the brain. A lot these findings are based on the study of the locomotion of a lamprey which is a primitive fish (in [3] an extensive review is presented). For instance, researchers

discovered [4] after extracting and isolating the spinal cord from the body, that the spinal cord, when excited with electrical stimulations, will produce fictive locomotion. This indicates that sensory information is not needed to generate such rhythmic patterns. However, it plays a crucial role in shaping the generated pattern to keep the coordination between the CPGs and the body. In [5] they demonstrated that a mechanically driven treadmill can induce a normal looking walking gait in a deprecated cat.

These findings suggest that locomotion can be represented by a hierarchy of modules which interact with each other on different timescales which simplifies high level control and at the same time allows fast action against perturbations. For example, slight irregularities in the terrain are compensated very fast by the morphology of the legs without the need of the brain to intervene. Larger irregularities are handled by a slower reflex motion of which the runner becomes finally conscious at the highest but slowest contextual level.

Biological research is often used to improve the abilities of robots. For example, in [6] a salamander robot is controlled by a network of CPGs imitating the spinal cord of a salamander. In this work we investigate a multi-timescale control hierarchy applied on a robot leg. Each of the layers in the proposed hierarchy uses a random dynamical system of which only the readout layer is trained (eg. Reservoir Computing systems [7]). On the lowest level, the fastest timescale, the passive compliance of the leg interacts with the environment. The leg is driven by a pattern generator which generates learned rhythmic motor signals and gets feedback from the rotary encoders in the leg. On the highest level, we use a controller which reacts to slower contextual changes. This hierarchy separates the motor commands from the functional control which is done by the high level controller. Furthermore, by using a Reservoir Computing network on each layer, we open the door for potentially rich motor skills and advanced control which is topic for future investigation.

The remainder of this paper is structured in six Sections. In Section 2 we start by giving a rough overview of the proposed hierarchical control scheme. Next, in Section 3 we elaborate more deeply on the core technique used to build our hierarchy. After that, the two main building blocks of our control hierarchy are discussed in more detail. Afterwards, the hierarchy is validated by means of three preliminary experiments in Section 6. Finally, we end this paper by giving our conclusions.

2 Proposed hierarchy

In all vertebrates, neural circuits located in the spinal cord can be found that are responsible for generating rhythmic activations used for locomotion. These neural circuits are called Central Pattern Generators (CPGs) and are currently modeled by roboticists to control robot locomotion. One of the key advantages that can be identified is that these CPGs typically have only a few control parameters and thus reduce the control problem [3].

In this work we propose the use of a hierarchical (artificial) neural system for adaptive locomotion control. This system, illustrated in Fig. 1(a), consists of two building blocks: a pattern generator and a controller. On the lowest level,

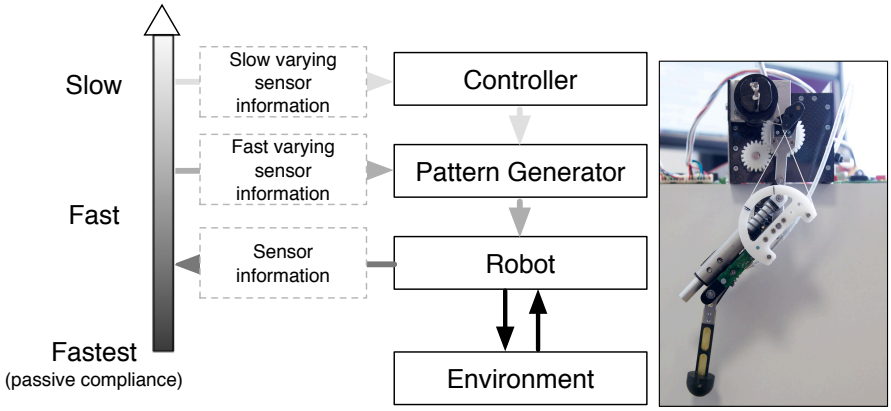


Fig. 1. (a) We present an overview of the control hierarchy. Our approach uses two building modules working on different time scales. The first module, the pattern generator, operates at a fast time scale and gets feedback from fast varying sensors. The parameters of the pattern generator are adapted by a controller, operating at a slower time scale, which gets feedback about slowly variating motion properties. The environment is included to illustrate that very fast perturbations caused by interacting with the environment are handled by the passive compliance of the leg. (b) Shows the actual leg of the Oncilla robot build in the AMARSi consortium on which the experiments were performed.

a pattern generator operates at a fast time scale and embeds a learned rhythmic pattern which is given to the motors of the robot. The rotary encoders of the motor system provide the pattern generator with direct feedback. Only environmental changes which are are unhandleable by the passive compliance of the leg, will be visible for this encoder. On the highest level, and thus slower time scale, the controller tunes the parameters of the pattern generator online in such a way that it keeps track of the slow varying parameters of the resulting motor. To achieve this, the sensor information presented to the controller is preprocessed by calculating for example amplitude and offset. As a result, the proposed hierarchy operates at multiple time scales which allows the use of a more advanced controller (which often acts slower) while the pattern generator can be kept relatively simple and can immediately act on for instance perturbations. In other words, the motor commands generated by the pattern generator are separated from the functional control which is done on a higher level.

3 Reservoir Computing

The core technique used for each of the building blocks in our hierarchical control approach is Reservoir Computing (RC). This is a collection of efficient training methods for random dynamical systems in which only the readout weights are trained [7]. In the past, RC has been independently introduced as Echo State

Networks [8] and Liquid State Machines [9]. In previous work, RC has already proven its capabilities in a broad range of applications including robot localization [10], chaotic time series prediction [11] and speech recognition [12]. Additionally, researchers are making efforts to directly implement such systems on hardware [13].

The most commonly used flavor of RC is the Echo State Network approach which uses a random recurrent neural network of sigmoidal neurons. Training such a system starts by randomly creating the weight matrices $\mathbf{W}^{\text{res}}_{\text{res}}$, $\mathbf{W}^{\text{res}}_{\text{inp}}$, $\mathbf{W}^{\text{res}}_{\text{out}}$ and $\mathbf{W}^{\text{res}}_{\text{bias}}$ (these weights are usually drawn from a uniform or a normal distribution) which respectively determine reservoir-to-reservoir, input-to-reservoir, output feedback and bias weights. The reservoir weight matrix $\mathbf{W}^{\text{res}}_{\text{res}}$ is typically scaled such that the spectral radius, e.g. the largest eigenvalue, is smaller than 1. This guarantees that the entire system is operating at the edge of chaos, where its computational power is greatest [14]. Some learning paradigms, such as FORCE learning [15], require that the spectral radius is larger than 1 as long as additional inputs are able to restrain the system's dynamics.

After constructing the weight matrices the network can be simulated. Therefore, every time step, the neuron states are updated using the following equation:

$$\begin{aligned}\mathbf{x}[k+1] = (1-\lambda)\mathbf{x}[k] + \\ \lambda \tanh\Big(\mathbf{W}^{\text{res}}_{\text{res}}\mathbf{x}[k] + \mathbf{W}^{\text{res}}_{\text{inp}}\mathbf{u}[k] + \mathbf{W}^{\text{res}}_{\text{out}}\mathbf{y}[k] + \mathbf{W}^{\text{res}}_{\text{bias}}\Big).\end{aligned} \tag{1}$$

The states $\mathbf{x}[k+1]$ at time step $k+1$ depend on the previous states $\mathbf{x}[k]$, input $\mathbf{u}[k]$, a bias and the (optional) output feedback of the system $\mathbf{y}[k]$. By changing the leak-rate λ, the system's dynamics can be tuned effectively [8].

For training the output weights $\mathbf{W}^{\text{out}}_{\text{res}}$ a learning algorithms is needed that rapidly reduces the difference between the actual and desired output, and keep it small while converging to a set of fixed output weights. The resulting weights maintain a small error without further modification [15]. Recursive Least Squares (RLS) is one of those learning rules that satisfy all conditions for FORCE learning. During training, the reservoir states are updated according to equation 1 while at every time step the readout weights $\mathbf{W}^{\text{out}}_{\text{res}}$ and the output $\mathbf{y}[k+1]$ are adjusted according following RLS equations:

$$\mathbf{y}[k+1] = \mathbf{W}^{\text{out}}_{\text{res}}\mathbf{x}[k+1] \tag{2}$$

$$\mathbf{e}[k+1] = \mathbf{y}[k+1] - \mathbf{y}_{\mathbf{desired}}[k+1] \tag{3}$$

$$\mathbf{P} = \mathbf{P} - \frac{\mathbf{P}\mathbf{x}[k+1]\mathbf{x}^{\mathbf{T}}[k+1]\mathbf{P}}{1 + \mathbf{x}^{\mathbf{T}}[k+1]\mathbf{P}\mathbf{x}[k+1]} \tag{4}$$

$$\mathbf{W}^{\text{res}}_{\text{out}} = \mathbf{W}^{\text{res}}_{\text{out}} - \mathbf{e}[k+1]\mathbf{P}\mathbf{x}[k+1]. \tag{5}$$

Here $\mathbf{e}[k+1]$ is the error at time step $k+1$ and $\mathbf{P}$ is an estimation of the inverse of the correlation matrix. After training, the weights $\mathbf{W}^{\text{res}}_{\text{out}}$ are kept fixed and the system can be used. However, when online adaptation is necessary, it is possible to train continuously.

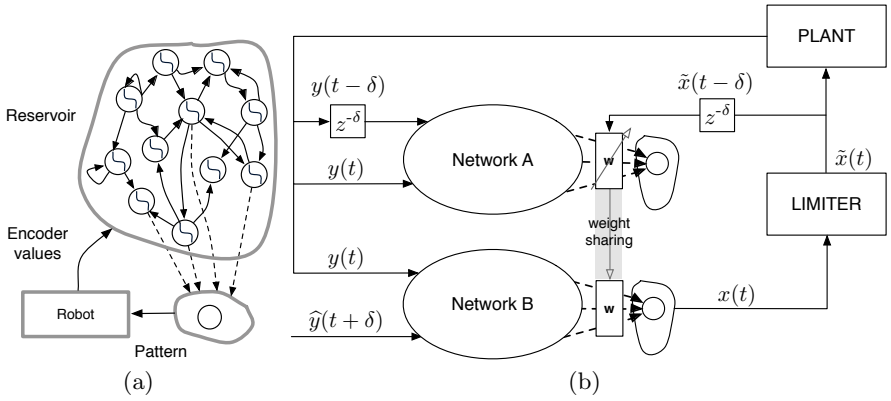


Fig. 2. The two main building blocks of our hierarchical approach. On the left (a) one can see a schematic overview of a Reservoir Computing based pattern generator. The rotary encoders in the robot leg are used for feedback. On the right (b) an overview of the controller we use to modulate the pattern generator (the plant) is shown.

4 Modulatable Pattern Generator (MPG)

As we discussed in Section 2 the robot is controlled directly by a pattern generator which is able to imitate demonstrated rhythmic motions. The pattern generator is implemented in a RC-network. In previous work we showed that by using additional inputs, the generated patterns can be modulated [16]. This work was extended in [17] by adding the capability to encode discrete and rhythmic motion patterns into a single recurrent neural network as respectively a limit cycle and a fixed point attractor. Typical parameters that are used for this are summarized in Table 1. More recently, in [18] a new method to modulate the shape of a learned rhythmical pattern was illustrated based on tuning the bias weights of the neurons instead of using additional inputs. In summary: the influence of adding a small bias to each neuron is determined after training. Therefore, each neuron is perturbed separately with a small constant bias. After perturbing each neuron, one can observe the influence of this on the properties of the output signal. For each property that one wants to track, a control vector can be composed which can be used to modulate the output signal. In this work we will use this modulation approach to change the properties of the generated signal. We only determine the control vector that influences the signal amplitude and offset. More complex property transformations will be shown in future work.

5 Controller

In [19] and [20] we introduced a novel feedback controller which learns to control a plant (dynamical system) by online learning an inverse plant model based on real-time controlled plant-input/output pairs. In parallel, this preliminary model is used to actually control the system, producing a new plant-input output pair

Table 1. Summary of all parameters in a pattern generating RC-network with their typical values.

Parameter	Description	Value
N	number of neurons	100 to 2000
λ	leak-rate	0.01 to 1
ρ	spectral radius	0.99 to 1.5
β	bias weight variance	0 to 1
ω	output feedback scale	0 to 10
α	learning rate, only FORCE learning case, determines initialization of $\mathbf{P_{init}} = \frac{\mathbf{I}}{\alpha}$	0.1

which gradually improves the inverse model. At the core of this feedback controller we use a RC-network to accommodate the inverse model. However, as described in [20], any dynamical system with a high dimensional state representation can be used to accommodate such model as well. In this paper we apply the same feedback controller (shown in Fig. 2(b)) to modify the bias vector to MPG neurons, which are sensitive to the amplitude and offset of the generated motion. Although the MPG is fully responsible for the produced motion, the controller allows the MPG to track a desired amplitude and offset which are changing more slowly compared to the position of the motor.

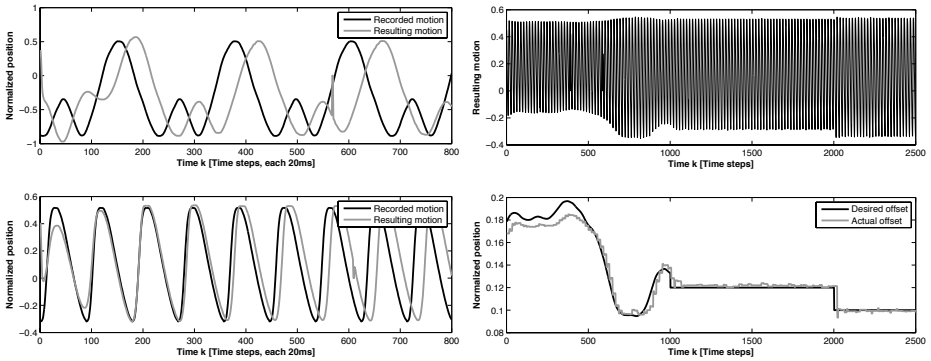
As shown in Fig. 2(b), the feedback controller uses two identical RNN of which only the inputs differ. The output weights of Network A are trained (applying RLS) by observing plant-input/output pairs $((y(t-\delta), y(t)), (x(t-\delta)))$. These output weights are used by Network B to produce a plant-input $x(t)$ given the actual and desired plant-output, $y(t)$ and $\hat{y}(t+\delta)$ respectively.

6 Experiments

In this section we apply the discussed control hierarchy on a prototype robot leg (of the Oncilla robot platform) which is developed in the AMARSi consortium and shown in Fig. 1(b). This robot leg is controlled by a motor control board which in turn is driven by a small computer. However, because of the computational limitations of this onboard computer and to ensure the desired communication timings, all calculations are offloaded to a much more suited computational unit. In this work we want to demonstrate the above described control hierarchy concept on a simple task. Although the control of multiple servos is possible, we will limit the amount of controlled servos to 1. The robot leg is controlled by a simple P-controller which converts the positions, generated by the MPG, to a torque signal. However, to allow for changes in the robot dynamics to be visible in its motion, the used P-parameter is smaller than optimal and the amount of torque is limited. In Table 2 we show the associated parameters concerning the used feedback controller and MPG setup. Additionally, the different timings are given at which each system is interacting with another system. As mentioned before, the used feedback controller is interacting at a much slower rate compared to the MPG's control rate.

Table 2. Summary of all our setup parameters used in the experiments.

Parameter	Pattern generator	Controller
N	500	500
λ	0.14	1.
ρ	1.4	1.
α	0.1	0.01
time scale	20ms	100ms
input scaling	1.0	0.1
β	0.5	0.5

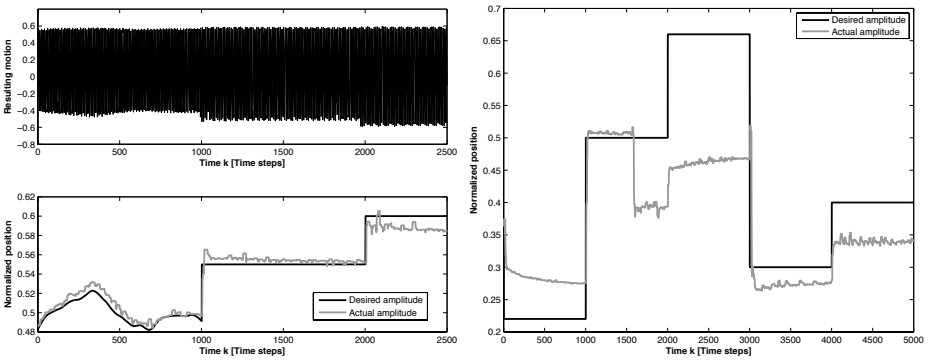


(a) (b)

Fig. 3. (a) Shows two different recorded motions together with the actual reproduction by the robot leg. (b) Depicts the actual motion during offset control (top) together with the desired and actual offset (bottom).

6.1 Learning by Imitation

By limiting the servo's torque the robot leg can be back-driven, allowing the demonstration of a given motion. In this work we impose a mixed sinusoidal motion which afterwards is used to train the MPG-network. When training is completed, the necessary gradient vectors to the MPG-neurons that affect the amplitude and offset, are computed. In Fig. 3(a) the actual trained leg motion is shown for two different imposed patterns which are shown as well. The first pattern is a mixed sine pattern while the second motion is a sinusoidal pattern with a slower and faster phase. The latter is similar as in a swing/stance phase gait. Both resulting motions show a phase shift caused by integrating the robot leg into the feedback loop. A change in the generated pattern has to propagate through the dynamics of the robot leg, before the correct leg angle is visible for the leg encoder. Because of the feedback loop, this variating delay eventually affects the generated pattern. This illustrates that a higher level control is necessary to modulate the pattern generator such that these dynamics are taken into account.



(a) (b)

Fig. 4. (a) Depicts the actual motion during amplitude control (top) together with the desired and actual amplitude (bottom). (b) This plot illustrates how the proposed hierarchy reacts to changes in the dynamics of the robot leg during amplitude control.

6.2 Motion Modulation by Controlling the MPG

After learning the recorded motion, the feedback controller is applied to modulate the amplitude and offset of the motion, which are only observable on a slower time scale. As mentioned before, this can be achieved by controlling the bias of amplitude/offset responsive MPG-neurons. Fig. 3(b) shows the desired and actual offset which is controlled by the feedback controller. To control the offset, the highest level of our proposed hierarchy was interacting with the MPG every 100 ms (5 times slower than the interaction rate of the MPG). Fig. 4(a) demonstrates a similar experiment but for amplitude control. Additionally, the actual resulting positions are depicted at the bottom of both Fig. 3(b) and 4(a).

6.3 Adapting to Changes in Robot Dynamics

In the previous experiment we showed that the generated motion can be modulated. However, we want to investigate the capability of the proposed hierarchy to adapt to changes in the dynamics of the robot or in its environment. In our experimental setup we are limited in introducing changes to adding weight to the robot leg. We increase its mass by hanging an extra weight at the tip of the leg. As a result, the amplitude of the motion will be reduced and the offset will move closer to the lowest point of the leg. However, this switch in dynamics will cause the inverse model of the feedback controller to adapt to these changes as well. As a result, during amplitude control the amplitude will eventually converge again to its desired value. In Fig 4(b) after 1500 time steps a mass of 100 g is added to the leg. After adjusting its internal model, the controller start compensating for the extra weight at time step 3000 by controlling the bias of the MPG.

7 Conclusions

Roboticists are often inspired by biology to improve the abilities of robots. One of the main directions is the use of central pattern generators which reduce the dimensionality of the locomotion control problem. Inspired by this, we proposed the use of a multi-timescale hierarchical controller that uses random dynamical systems for each layer. On the lowest level, a pattern generator is able to embed any rhythmic signal. This pattern generator interacts directly with the motor of the leg on a fast timescale. On the highest level, and thus slowest time scale, a controller tunes the parameters of the pattern generator online such that it keeps track of the slow varying parameters of the resulting motion. To achieve this, the sensor information presented to the controller is preprocessed by calculating for example the amplitude and offset. Since the controller acts on a slower timescale, this controller can be very advanced and might consist of a very large random dynamical system. On the other hand, the pattern generator is fast enough to react immediately on small perturbations which can not be compensated by the morphology of the robot (passive compliance).

By means of three preliminary experiments on the AMARSi Oncilla leg, we validated the proposed control hierarchy. In a first experiment we showed that the hierarchy is able to capture a (by hand) shown rhythmic motion pattern which is embedded by the pattern generator. In the second experiment we illustrate that the higher level controller is able to track slow varying properties such as amplitude and offset, by only controlling the bias of the pattern generator. Finally, in the third experiment, we demonstrated that the control hierarchy is able to deal with new situations such as changes of the leg weight.

We are now planning to apply the proposed control hierarchy on the in the AMARSi consortium developed quadruped Oncilla robot. We will mainly try to tackle two problems: (1) controlling the stability of the robot by considering a measure of stability as the slow control property to track, and (2) we will embed multiple gaits for the Oncilla robot.

Acknowledgment

This work was partially funded by a Ph.D. grant of the Institute for the Promotion of Innovation through Science and Technology in Flanders (IWT-Vlaanderen) and the FP7 funded AMARSi EU project under grant agreement FP7-248311.

References

1. S. Kiebel, J. Daunizeau, and K. Friston, "A hierarchy of time-scales and the brain," *PLoS computational biology*, vol. 4, no. 11, p. e1000209, 2008.
2. P. Stein, *Neurons, networks, and motor behavior*. The MIT Press, 1999.
3. A. Ijspeert, "Central pattern generators for locomotion control in animals and robots: a review," *Neural Networks*, vol. 21, no. 4, pp. 642–653, 2008.
4. A. Cohen and P. Wallen, "The neuronal correlate of locomotion in fish," *Experimental brain research*, vol. 41, no. 1, pp. 11–18, 1980.

5. S. Rossignol, "Locomotion and its recovery after spinal injury," *Current opinion in neurobiology*, vol. 10, no. 6, pp. 708–716, 2000.
6. A. Ijspeert, A. Crespi, D. Ryczko, and J. Cabelguen, "From swimming to walking with a salamander robot driven by a spinal cord model," *Science*, vol. 315, no. 5817, pp. 1416–1420, 2007.
7. D. Verstraeten, B. Schrauwen, M. D'Haene, and D. Stroobandt, "An experimental unification of reservoir computing methods," *Neural Networks*, vol. 20, pp. 391–403, 2007.
8. H. Jaeger, "The "echo state" approach to analysing and training recurrent neural networks," German National Research Center for Information Technology, Tech. Rep. GMD Report 148, 2001.
9. W. Maass, T. Natschläger, and H. Markram, "Real-time computing without stable states: A new framework for neural computation based on perturbations," *Neural Computation*, vol. 14, no. 11, pp. 2531–2560, 2002.
10. E. A. Antonelo, B. Schrauwen, and D. Stroobandt, "Event detection and localization for small mobile robots using reservoir computing," *Neural Networks*, vol. 21, pp. 862–871, 2008.
11. H. Jaeger and H. Haas, "Harnessing nonlinearity: predicting chaotic systems and saving energy in wireless telecommunication," *Science*, vol. 308, pp. 78–80, 2004.
12. B. Schrauwen, M. D'Haene, D. Verstraeten, and J. Van Campenhout, "Compact hardware liquid state machines on fpga for real-time speech recognition," *Neural Networks*, vol. 21, pp. 511–523, 2008.
13. Y. Paquot, F. Duport, A. Smerieri, J. Dambre, B. Schrauwen, M. Haelterman, and S. Massar, "Optoelectronic reservoir computing," *SCIENTIFIC REPORTS*, vol. 2, pp. 1–6, 2012.
14. R. A. Legenstein and W. Maass, "Edge of chaos and prediction of computational performance for neural microcircuit models," *Neural Networks*, pp. 323–333, 2007.
15. D. Sussillo and L. Abbott, "Generating coherent patterns of activity from chaotic neural networks," *Neuron*, vol. 63, no. 4, pp. 544–557, 2009.
16. F. Wyffels and B. Schrauwen, "Design of a central pattern generator using reservoir computing for learning human motion," in *Proceedings of the ECSIS Symposium on Advanced Technologies for Enhanced Quality of Life*, 2009, pp. 118–122.
17. T. Waegeman, F. wyffels, and B. Schrauwen, "A discrete/rhythmic pattern generating rnn," in *Proceedings of the 20th European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning.* Louvain-la-Neuve, Belgium: Ciaco - i6doc.com, 2012, pp. 567–572.
18. J. Li and H. Jaeger, "Minimal energy control of an esn pattern generator," Jacobs University, Tech. Rep., 2011.
19. T. Waegeman and B. Schrauwen, "Towards learning inverse kinematics with a neural network based tracking controller," in *Neural Information Processing.* Springer, 2011, pp. 441–448.
20. T. Waegeman, F. Wyffels, and B. Schrauwen, "Feedback control by online learning an inverse model," *IEEE Transactions on Neural Networks and Learning Systems (submitted).*